

by Bruce A. Bergman

Process Improvements

This month, I'm going to get off track a bit and write about engineering process improvements, a topic that many people are interested in, yet is still somewhat of a black hole. Instead of the in-depth discussion that should take place, I'll just give an overview of the topic and if you feel that you would like to hear more about it, please let me know.

Engineering process improvements are tangible activities that engineering departments can perform that *should* improve productivity and products. In other words, if done correctly, the engineers in that department should become more productive, more efficient, more knowledgeable, and, at the same time, contribute to the overall success of the department by producing items that work better, read better, and so on. I use the word "items" loosely because engineering process improvements are not limited to software, although that application is the most common. They can be applied to documentation, hardware, and so on. The end result, which is to improve upon what existed before, should be the same. The engineering process, a catchall for the activities from conception to completion, can always be improved.

Engineering process improvements are not a new topic in Computer-Science Land. Scientists have been working on these improvements since day one, and much of the benefit we enjoy today is because of the years of research they have done within their own engineering departments. The overall concept is simple. Implementing it *successfully* is not. The years have shown this fact over and over, and, hopefully, we can learn from scientists who have taken the rough road.

SIMPLE ENHANCEMENTS

Generally speaking, process improvements address several different thrusts

Design meetings should be held at least twice during the design cycle: before the design begins and after the design has been completed.

at engineering activities. Some of the more common improvements to be implemented include requirements reviews, design meetings, coding guidelines, style guidelines, code inspections, tiered testing, and product verification.

The one thing these improvements have in common is that they look easy to implement but, in reality, can often be nightmares. For example, take style guidelines. Most engineers have developed their own style for coding or writing, and to be given a document describing, in minute detail, exactly where all those matching braces should be placed is tantamount to asking for hate email to come flooding through your network.

In general, process improvements would be manifested under the greater umbrella of quality assurance. In small companies, however, quality assurance is often an activity that is done by no one and yet by everyone. Either way, these improvements can be implemented successfully by any company.

REQUIREMENTS AND DESIGN

It is important to match a customer's requirements with the potential for completion by the engineering depart-

ment. We do this with a requirements review. Basically, this means reviewing a list of requirements and deciding if meeting those requirements is feasible. Some engineering departments like to assign a level-of-effort value to each requirement to develop a good picture of the overall project. Some engineering activities may not have requirements reviews at all.

Design meetings are used in the early stages of the development cycle to attempt to catch design flaws, plan future action, identify the development course, and so on. These design meetings come in a variety of flavors and can be as structured as the preliminary and critical design reviews, or as unstructured as an informal meeting of the designer and a couple of engineers. Studies show that if design meetings are held before moving on to later stages in the development cycle, as many as 85% of all problems can be found.

Design meetings should be held at least twice during the design cycle: before the design begins and after the design has been completed. Of course, *all* design meetings held between these points decrease the chance of significant rework if a problem is identified. At the first design meeting, participants should identify the goal of the design. What is the ultimate product of the design? What should be produced after the design is done? When the goal is clear, the participants should identify potential problem areas, brainstorm for ideas, and so on. The participants should refrain from trying to solve the problems or do the actual design. The idea behind this first meeting is to plot future courses of action.

Later meetings should be used to present a part of the design or address specific concerns that come up during the design process. Ultimately, when the design is done, review the entire design for omissions, problems, complete-

ness, and so on. Perhaps a check against the requirements matrix is needed.

The next stage is to do the actual work, which, in the software context, means coding. In the hardware context, it might mean actually building the product. In a documentation context, it might be writing a user's guide. Regardless of what it is applied to, this stage of the development can apply other engineering process improvements that have been developed. That same

technical writer who is about to embark on creating the user's guide can apply the style guidelines that have been developed to produce documentation that meets the department's standards.

One important aspect at this stage of development is the implementation of code inspections or reviews. Again, I'll talk about this concept in the software context, but it is just as applicable elsewhere. In fact, the person credited for the creation of this concept is Michael

Some of the benefits of code reviews are that they assist in the transfer of knowledge from engineer to engineer.

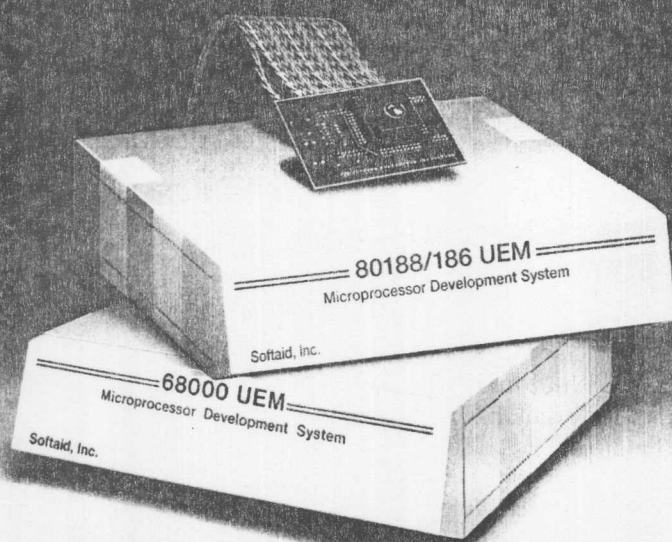
Fagan, who was working for IBM back in the mid-1970s. While code inspections today are used primarily for software, Fagan intended his concept to be used on hardware. Reviews can be helpful in several areas of software or hardware.

CODE INSPECTIONS

Code inspections and peer reviews can be divided into two categories: code reviews and code walk-throughs. The primary difference is the level of formality. Code reviews are intended to be formal, held at logical transition points in the development cycle, and provide metrics for the object being reviewed. Code walk-throughs are less formal and are meant to touch base with other developers and individuals to get a feel for how the project is progressing.

Some of the benefits of code reviews are that they assist in the transfer of knowledge from engineer to engineer. They also allow for more accurate project schedules and estimations, provide metrics for problem analysis, product development, and so on. Similar to design meetings, code reviews are held with others in attendance, a specific goal for the meeting, a predetermined list of invitees, and an exit criteria. After each review, and especially if it is a formal review, the group should decide on an exit criteria. That is, should the development continue? Are there minor problems that should not hold up devel-

The 16-Bit Solution



Affordable Performance for the 68000 & 80186

Price and performance are no longer mutually exclusive. Softaid's 16-bit UEM Emulators find bugs *fast*. And only Softaid has the Fiber-Optic Link that reduces download time to virtually zero. Spend a lot less time waiting, and a lot more time debugging.

All of Softaid's UEM Emulators include built-in performance analysis and a source-level debugger for C, C++ and PL/M. Our real-time Memory Access Monitor instantly captures

"stack creep" and those programs that mysteriously "wander off."

Collect real-time trace data with pre, post, or center triggers. Display it in C source or intermixed with the

disassembled trace data. Or, use your own clock to collect logic analyzer data.

Is there a better solution than Softaid? At prices from \$6,495 to \$7,495, this may be the best 16-bits of advice you ever get.

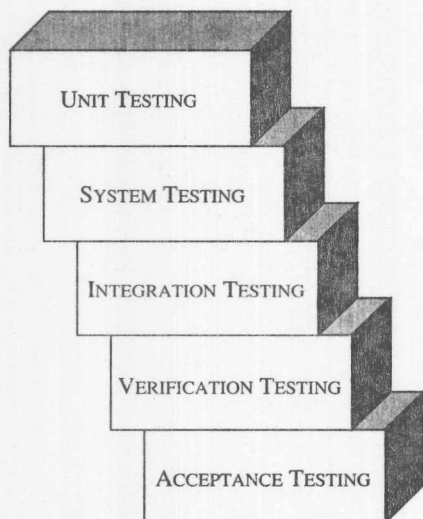
Processor	Speed
80286	16MHz
80188/C188	16MHz
80186/C186	16MHz
80188EB/186EB	16MHz
68000/8/10	16MHz
68302	16MHz
8088/86	10MHz
V20/30	16MHz
V40/50	12.5MHz

8300 Guilford Rd. • Columbia, MD 21046
(800) 433-8812 • FAX: (301) 381-3253

SOFTAID, Inc.

CIRCLE #209 ON READER SERVICE CARD

Figure 1
Tiered testing.



opment, but that should be fixed? Or are there major problems that need to be addressed before continuing? Engineers who participate in code inspections are more than twice as efficient as they would be if they found the same problems in later stages such as unit testing.

Remember, these meetings should not be used to solve problems. Rather, they should be used to *identify* problems. If this focus is kept, the meetings go much more smoothly.

TESTING

Tiered testing is an important part of any engineering process. The idea behind tiered testing is to divide testing into several distinct phases that complement and overlap each other to some degree. If done properly, this method produces a tiered effect, as shown in Figure 1.

At the earliest stage, unit testing is done on some small, logically removed unit of the finished product. In software, this unit might be a module, package, or reusable object, while in larger interests, a unit might be a component of the final product. The idea is to apply a set of limited (but extensive) tests to one small part while exercising as much of the unit as possible. Any problems

encountered can be dealt with at the unit level.

In system testing, a larger group of units are tested together to make sure they work well as a group. This form of testing is helpful in identifying intermodule problems, interface problems, and so on. This type of testing is slightly different from integration testing, which really should be done on a larger scale. Integration testing, for instance, might test the way two components act when

they are used together, which should identify problems such as where your standard output went if it didn't make it through the pipe successfully. ("It goes to never-never land....")

Verification testing is used to show that the finished product does what it was intended to do. This stage of testing can be done when the product, as a whole, is finished and ready to ship. A logical part to review in this stage is the installation procedure. If a customer re-

68331, 68332, 68340 Developers: Don't Ignore Background Mode!

If you're ignoring Background Mode, you're ignoring price/performance, ease-of-use, and non-intrusiveness in your debugging system.

Motorola Doesn't.

Motorola realized the benefits of on-chip debugging. That's why they designed Background Mode into every CPU32 microcontroller.

Introducing the EST Series 300!

The EST Series 300 controls your CPU32 target exclusively through a 5 MHz Background Mode link. It's completely non-intrusive, since it doesn't require your target's RAM, ROM, interrupts, or serial port.

Packed with Debugging Power.

Of course, the EST series 300 delivers the features you need: simple and conditional breakpoints on RAM/ROM, data breakpoints, stepping, tracing, view/modify registers and memory, and system diagnostics, at a fraction of the price of a full-blown ICE.

Real-time Simulation.

To get a head start on your software, the EST Series 300 builds in a CPU32 controller and 128K of simulation RAM. Or, our Background Mode system works with your EVS board, right out of the box.

C Source Level Debugging.

EST has ported Intermetrics' XDB 5.0 to the Series 300. XDB delivers a proven windowed interface, complete source and symbolic information, C level tracing, and powerful macros.

Try Background Mode.

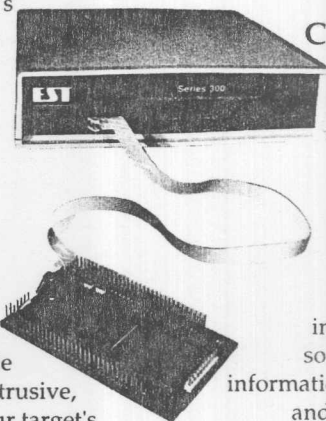
Just call us, and we'll send you an EST Series 300 for a *free* 14 day trial. No obligation. No Risk.

(617) 828-5588

**Embedded
Support
Tools Corp.**



10 Elmwood St., Canton, MA 02021



CIRCLE #210 ON READER SERVICE CARD

ceives a piece of software or hardware that can't be installed the way the documentation says it can, then verification testing was either omitted or done poorly.

Lastly, acceptance testing is a formal activity that seeks to prove that all requirements have been met. If the requirements review process is done properly, a requirements matrix should have been developed. This matrix is used in conjunction with specific tests to prove that the product performs as anticipated. This type of testing is wide ranging and might include hardware, software, and documentation. It might also include performance tests, visual inspections, assumed or implied completions, and so on.

The entire concept of engineering process improvements is exactly what the name says: to improve the engineering process. Improvements are achieved by applying logical and intelligent controls and checks to existing develop-

If the requirements review process is done properly, a requirements matrix should have been developed.

ment or new development. If everything is done correctly and instituted with respect for the individuals involved, the benefits should be self-evident after a period of time.

If you would like to learn more about

engineering process improvements or if you have successful ideas from your own organization, please drop me a line and let me know what you think. Opinions, tips, or suggestions can be sent to Embedded Systems Programming, 600 Harrison St., San Francisco, Calif. 94107, or use the Embedded Systems Programming Bulletin Board System at (415) 905-2689. We prefer tips to be as machine-independent as possible and primarily in C, Ada, Fort, or assembly language. Contributions I use will be published with appropriate credit and a small, but heartfelt, honorarium.

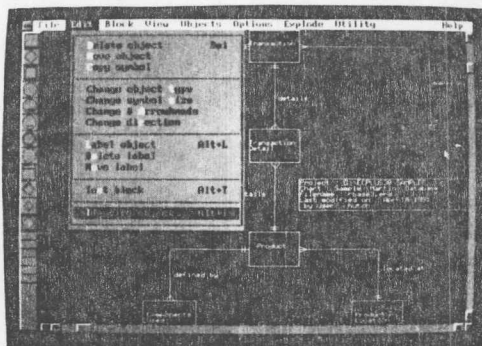
Bruce A. Bergman is a software engineer at Horizons Technology Inc. in San Diego, Calif. He welcomes contact and can be reached on CompuServe at 70431,3167, on his Bulletin Board System at (619) 268-9275, or on Usenet at fatcity!bruceb@crash.cts.com.

EasyCASE™ Plus

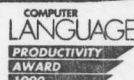
The affordable approach to software engineering... only \$495

NEW!
VERSION 3.0
Call for upgrade details.

Finally, there's a CASE tool that won't get in the way of your creativity... A tool that makes structured analysis, structured design and data modeling as easy as working with any other tool on your PC - EasyCASE Plus! Using EasyCASE Plus' new, easy to use graphical user interface (GUI), you'll be creating and editing charts, linking them, and building your data dictionary in no time. As well as being easy to use and easy to learn, EasyCASE Plus is easy on your budget! Ask any user. They'll tell you it's the best buy for your PC based CASE tool needs. Discover why over 4,000 software professionals use EasyCASE Plus and how you can join them!



"EasyCASE Plus is a well designed, low priced tool that is easy to learn and provides excellent diagramming capabilities... EasyCASE Plus is an excellent investment."



Features:

- IBM SAA/CUA compliant graphical user interface (GUI)
- Extensive diagram editing features
- Integrated dBASE III compatible data dictionary
- Integrated dictionary manager, reports manager, process editor
- Hierarchical chart linking & process decomposition
- Record and element definitions
- Extensive printer, plotter and desktop publishing support
- Data dictionary import, export, and merge
- On-line help
- Comprehensive documentation with tutorial
- Access to your database, word processor, DOS, etc.
- Integrated diagram analysis (optional)

Requirements:

Runs on: IBM PC or PS/2 (AT recommended), DOS 3.1 or higher, EGA/VGA color, mouse, 640 K RAM (500 K free), 1 MB EMS recommended, math co-processor supported. Printers/Plotters Supported: Epson FX & LQ, IBM Graphics & Proprinter X24, HP QuietJet, DeskJet, & LaserJet, HP Plotters, PostScript.

EasyCASE Professional \$649
(includes integrated DFD level balancing and data dictionary/diagram analysis)

Methods:

- Yourdon/DeMarco
- Gane & Sarson
- Ward-Mellor/Hatley
- Yourdon/Constantine
- Martin
- Chen, Bachman

Diagram Types:

- Data Flow Diagrams (DFDs)
- Structure Charts
- State Transition Diagrams
- Entity Relationship (ERDs)
- Data Model Diagrams
- Transformation Schema (real-time DFDs)

**Evergreen
CASE
Tools**

16650 NE 79th Street
Suite 200
Redmond, WA 98052
USA
FAX: (206) 883-7676

Call today for a brochure!
Tel: (206) 881-5149

©1991 by Evergreen CASE Tools, Inc., All Rights Reserved. All trademarks are the property of their respective companies.

CIRCLE #211 ON READER SERVICE CARD